OULUN AMMATTIKORKEAKOULU

Juho Koskelainen

VARASTONSEURANTAJÄRJESTELMÄ

VARASTONSEURANTAJÄRJESTELMÄ

Juho Koskelainen
Opinnäytetyö
Kevät 2015
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä(t): Juho Koskelainen
Opinnäytetyön nimi: Varastonseurantajärjestelmä
Työn ohjaaja(t): Timo Vainio
Työn valmistumislukukausi ja -vuosi: Kevät 2015 Sivumäärä: 20

Opinnäytetyön aiheena oli varastonseurantajärjestelmä tuotteiden seurantaan ja varastokirjanpitoon. Jokaisella tuoteyksiköllä on yksilöllinen viivakoodi jonka perusteella sen tulee olla yksilöllisesti jäljitettävissä, sekä järjestelmän tulee mahdollistaa tilaajalle pääsy tietokantaan etäyhteyttä käyttäen. Käyttöliittymän suunnittelussa tuli kiinnittää erityistä huomiota ohjelmiston helppokäyttöisyyteen ja selkeyteen.

Käyttöliittymä toteutettiin C#-ohjelmointikielellä .NET Frameworkia käyttäen ja datan tallentamiseen valikoitui tilaajan palvelimelle käyttöönotettava MySQL-relaatiotietokanta. Työ aloitettiin tietokannan suunnittelulla, jonka jälkeen toteutettiin varsinaisen varastonseurantajärjestelmän eri toiminnallisuuksien prototyypit.

Toteutuksen tuloksena saatiin toimiva varastonseurantajärjestelmän prototyyppi, joka otetaan käyttöön tilaajan toiminnassa ja jota jatkokehitetään testausvaiheen palautteen perusteella.

Asiasanat: varastokirjanpito, tietokannat, MySQL

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Engineering

Author(s): Juho Koskelainen
Title of thesis: Warehouse Tracking System
Supervisor(s): Timo Vainio
Term and year when the thesis was submitted: Spring 2015 Pages: 20

The goal of the thesis was designing and developing an inventory tracking system capable of tracking individual product items based on their unique bar code Ids. The system was required to enable giving the overseer ability to access the data via Internet connection.

The desktop application was created on .NET Framework using C# and the database was created on MySQL Server. The project was carried out by first implementing the database, followed by prototyping each of the required functionalities.

The client was supplied with a working prototype which will be developed furthermore based on the feedback of testing phase.

Keywords: inventory, databases, MySQL

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tavoitteena oli toteuttaa varastonseurantajärjestelmä helpottamaan tilaajan varastokirjanpitoa sekä lakisääteistä velvollisuutta tuotteiden jäljittämisestä. Vastuunalaisella varastonhoitajalla on oltava etäpääsy tietokantaan mahdollisiin tietopyyntöihin vastaamisen nopeuttamiseksi.

Aikaisemmin käytössä olleella järjestelmällä pidettiin kirjaa viivakoodeista tavaraa vastaanottaessa ja lähetettäessä asiakkaille. Tähän käytettiin kannettavaa viivakoodilukijaa, jonka data purettiin sellaisenaan tietokoneelle. Tämän järjestelmän puutteena oli se, ettei pelkkiä viivakoodeja tarkastelemalla saada täydellistä kuvaa varaston tilasta. Esimerkiksi saapuvat tuotteet luetaan usein lavakoodia käyttäen, mutta tuotteet voidaan lähettää eteenpäin lavasta purettuina laatikoina.

Periaatteessa lakisääteinen velvollisuus tuotteiden jäljitettävyydestä täyttyi toki vanhallakin järjestelmällä, mutta viranomaisilta tulevan mahdollisen tietopyynnön sattuessa tilaajan olisi pahimmillaan huomattavan työlästä jäljittää pyydetty viivakoodi. Lisäksi edellä mainitusta syystä varastokirjanpidoksi vanhasta järjestelmästä ei ollut.

Järjestelmän tuli siis toimia siten, että työntekijä kykenee pelkkää viivakoodilukijaa käyttäen pitämään varastokirjanpidon ajan tasalla. Tämä vaati käytännössä relaatiotietokantaa, johon tuotteet tallennetaan yksiköittäin. Kun esimerkiksi lava vastaanotetaan järjestelmällä tulee olla tiedossa jokaisen lavalla kuljetetun laatikon ja jopa laatikoiden sisältämien yksiköiden koodit. Tämä tekisi tuotteesta riippuen jopa tuhat tietuetta yhtä vastaanotettua lavaa kohti, joten automatisointi oli tarpeen.

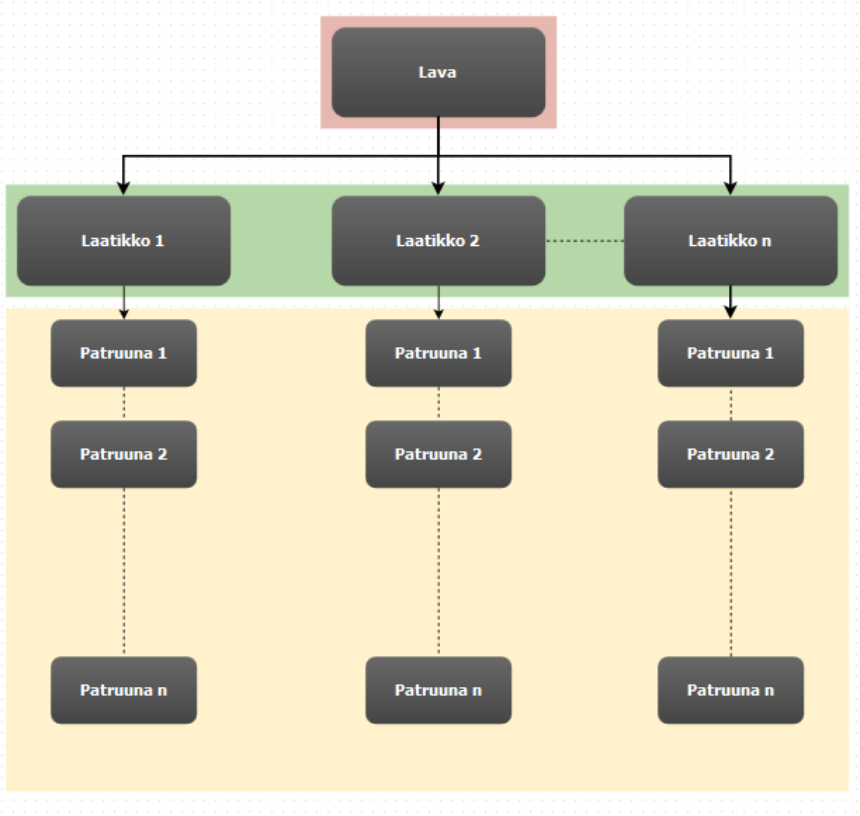
Oma täydellinen tietokanta mahdollistaa myös sähköisten lähetyslistojen luomisen asiakkaille, joihin vaatimus tuotteiden jäljittämisen mahdollistavasta kirjanpidosta laajentuu toukokuussa 2015.

Työn tavoitteiksi rajattiin tietokannan ja työpöytäsovelluksen prototyypin toteuttaminen kirjanpitoa varten. Etäpääsy tietokantaan toteutettaisiin opinnäytetyön valmistumisen jälkeen jatkokehityksessä.

2 SUUNNITTELU

Varastonseurantajärjestelmän tehtävänä oli siis täydellinen varastokirjanpito viivakoodien perusteella. Jokaisesta tuotteesta tuli olla jälkikäteen varmennettavissa, mille asiakkaalle se on toimitettu.

Seurattavat tuotteet kuljetetaan pääsääntöisesti lavoihin (kuva 1). Jokaisella lavalla, laatikolla ja patruunalla on oma yksilöllinen viivakoodinsa, mutta koodin perusteella ei voida luotettavasti päätellä yksiköiden mahdollisesti sisällään pitämien tuotteiden lukumäärää tai koodeja. Tämä muodostui ensimmäisen varsinaisen ongelman järjestelmää suunnitellessa: tuotteita vastaanottaessa olisi käytännössä mahdotonta purkaa pakkauksia niin, että kaikki koodit voitaisiin kirjata talteen. Kyseessä olisi tilaajan toimintaa kohtuuttomasti hankaloittava vaatimus.



KUVA 1. Seurattavien tuotteiden esimerkkikaavio

2.1 FEEML XML -standardi

Edellä kuvattu ongelma ratkaistiin ottamalla tavarantoimittajan kanssa käyttöön FEEM:n (Federation of European Explosive Manufacturers) standardisoima sähköinen lähetyslista. XML-muotoinen dokumentti tarjoaa standardoidun, ennustettavan rakenteen, joiden avulla liikekumppanit voivat siirtää dataa nopeasti, tehokkaasti ja tarkasti. FEEM suosittaa lähetysten sisällön ja pakkaushierarkian lähettämistä kaikille kuljetusketjun osapuolille, sillä jokainen siirto edellyttää kaikkien tuoteyksiköiden datan tallentamista. (1, s. 18)

XML eli Extensible Markup Language on tekstimuotoinen merkintäkielistandardi, joka mahdollistaa tiedon merkityksen kuvaamisen datan yhteydessä (2, s. 1). XML-dokumentti muodostuu elementeistä, joille on asetettu nimettyjä attribuutteja. Elementtejä voidaan sijoittaa toisten elementtien sisään muodostaen rakenteellisen kuvan tallennettavasta datasta, tässä tapauksessa tuotelähetyksessä siirrettävistä pakkauksista (kuva 2).

```
  <SummaryItem SID="S2" PSN="DE123">
    <ProducerProductCode>testproduct2</ProducerProductCode>
    <PurchaseOrderLineNumber>2</PurchaseOrderLineNumber>
    <DeliveryNoteLineNumber>2</DeliveryNoteLineNumber>
    <PackagingLevel>00</PackagingLevel>
    <ProductionDate>2012-07-20+02:00</ProductionDate>
    <NEW>0.001</NEW>
  </SummaryItem>
</SummaryItems>
<Units>
  <Unit PSN="DE123" UID="XYZ-001000000000">
    <ItemQuantity>240</ItemQuantity>
    <CountOfTradeUnits>6</CountOfTradeUnits>
    <PackagingLevel>04</PackagingLevel>
    <Units>
      <Unit PSN="DE123" UID="XYZ-001001000000">
        <ItemQuantity>40</ItemQuantity>
        <CountOfTradeUnits>4</CountOfTradeUnits>
        <PackagingLevel>03</PackagingLevel>
        <Units>
          <Unit PSN="DE123" UID="XYZ-001001001000">
            <ItemQuantity>10</ItemQuantity>
            <CountOfTradeUnits>10</CountOfTradeUnits>
            <PackagingLevel>01</PackagingLevel>
            <Items>
              <Item PSN="DE123" UID="XYZ-001001001001" SID="S1"/>
              <Item PSN="DE123" UID="XYZ-001001001002" SID="S1"/>
              <Item PSN="DE123" UID="XYZ-001001001003" SID="S1"/>
              <Item PSN="DE123" UID="XYZ-001001001004" SID="S1"/>
              <Item PSN="DE123" UID="XYZ-001001001005" SID="S1"/>
              <Item PSN="DE123" UID="XYZ-001001001006" SID="S1"/>
              <Item PSN="DE123" UID="XYZ-001001001007" SID="S1"/>
              <Item PSN="DE123" UID="XYZ-001001001008" SID="S1"/>
              <Item PSN="DE123" UID="XYZ-001001001009" SID="S1"/>
              <Item PSN="DE123" UID="XYZ-001001001010" SID="S1"/>
            </Items>
```

KUVA 2. Ote XML-esimerkistä (1, s. 22)

Seurantajärjestelmän vaatimuksiin lisättiin toiminto lähetyslistan lukemiseksi tietokantaan, jolloin saavutetaan täydellinen data vastaanotettavista tuotteista. Opinnäytetyössä lähetyslistan luku toteutetaan niin, että käyttäjä avaa dokumentin ohjelmassa manuaalisesti. Myöhemmin ohjelma voisi hakea lähetyslistat automaattisesti esimerkiksi sähköpostipalvelimelta.

2.2 Relaatiotietokanta

Datan tallentamiseen kaavailtiin ensimmäisistä palavereista alkaen MySQL-relaatiotietokantaa. Ohjaajani suositteli perehtymistä MySQL:n perustajajäsen Michael Wideniuksen luotsaamaan MariaDB-tietokantaan. Päädyimme kuitenkin käyttämään ennestään käytössä ollutta MySQL:ää. MariaDB on kuitenkin

varteenotettava vaihtoehto käyttöönotettavaksi myöhemminkin, sillä sen luvataan vastaavan MySQL-kirjastoja täydellisesti joten yhteensopivuusongelmia ei pitäisi tulla lainkaan (3). Lisäksi InnoDB-tietokantamoottori on korvattu XtraDB:llä, jonka tulisi suoriutua InnoDB:tä paremmin etenkin moniytimisillä suorittimilla (4).

Periaatteessa kaikkien pakkaustasojen koodit olisi voitu tietokannassa sijoittaa samaan tauluun, sillä lavojen ylemmän tason pakkausyksikköön viittauksen puuttumista lukuunottamatta tietueilla on samanlaiset muuttujat. Päädyin kuitenkin ratkaisuun, jossa tietokanta on jaettu kolmeen tauluun pakkaustason mukaan: lavat, laatikot ja patruunat. Normaalisti viivakoodeja käytetään vain lava- ja laatikkotasolla, ja epäilin tietokannan hakuaikojen kasvavan pitkässä juoksussa, mikäli myös patruunoiden tietueet sijaitsevat samassa taulussa. Laatikoiden patruunalukumäärästä johtuen pääkäytössä olevan taulun koko olisi kasvanut jopa 30-kertaiseksi ratkaisuuni verrattuna.

Viiteavaimet (foreign key) pitävät huolen siitä, että parent-child suhteet tuotteiden välillä ovat kunnossa. Viiteavaimen avulla esimerkiksi lavaa siirrettäessä myös lavaan kuuluvien laatikoiden ja niihin kuuluvien yksiköiden status päivittyy asiaankuuluvasti. Viiteavaimet voivat myös ehkäistä virheitä kirjanpidossa, sillä viiteavain estää mm. yksikön poistamisen, jos sen sisälle on kirjattu muita yksiköitä.

3 KÄYTETYT OHJELMISTOT JA LAITTEET

3.1 Microsoft Visual C# 2010 Express

Koska työpöytäsovelluksen kohteena olivat Windows-käyttöjärjestelmällä varustetut tietokoneet, oli entuudestaan tuttu ja ilmainen Microsoft Visual C# 2010 Express luonnollinen valinta kehitysympäristöksi. Ohjelmistokustannukset pidettiin minimissä eikä aikaa tarvinnut käyttää uuden kehitysympäristön opetteluun.

3.2 MySQL Workbench 6.0 CE

Myös MySQL Workbench oli tietokantoja käsitelleiltä kursseilta tuttu työkalu. Workbench mahdollistaa tietokannan luomisen varsin suoraviivaisesti EER-kaavion (Extended Entity-Relationship model) avulla. Graafinen malli auttaa hahmottamaan suunniteltavan tietokannan ja valmiin kaavion voi ajaa tietokantapalvelimeen Forward Engineer -toiminnolla muutaman napin painalluksella, useimmiten moitteitta.

3.3 CipherLab 8200 -viivakoodilukija

Varastoliikenteen lukemisessa tilaajalla oli entuudestaan käytössä CipherLabin 8200-sarjan mobiilitietokone. Olemassa olevan vakio-ohjelman avulla laitteella voidaan lukea varastoon saapuvat sekä lähteviä tuotteita lukiessa asettaa niille asiakas- tai statusnumero, jonka perusteella työpöytäsovellus voi tulkita mikä tuote siirretään mihinkin kohteeseen tietokannassa.

4 TOTEUTUS

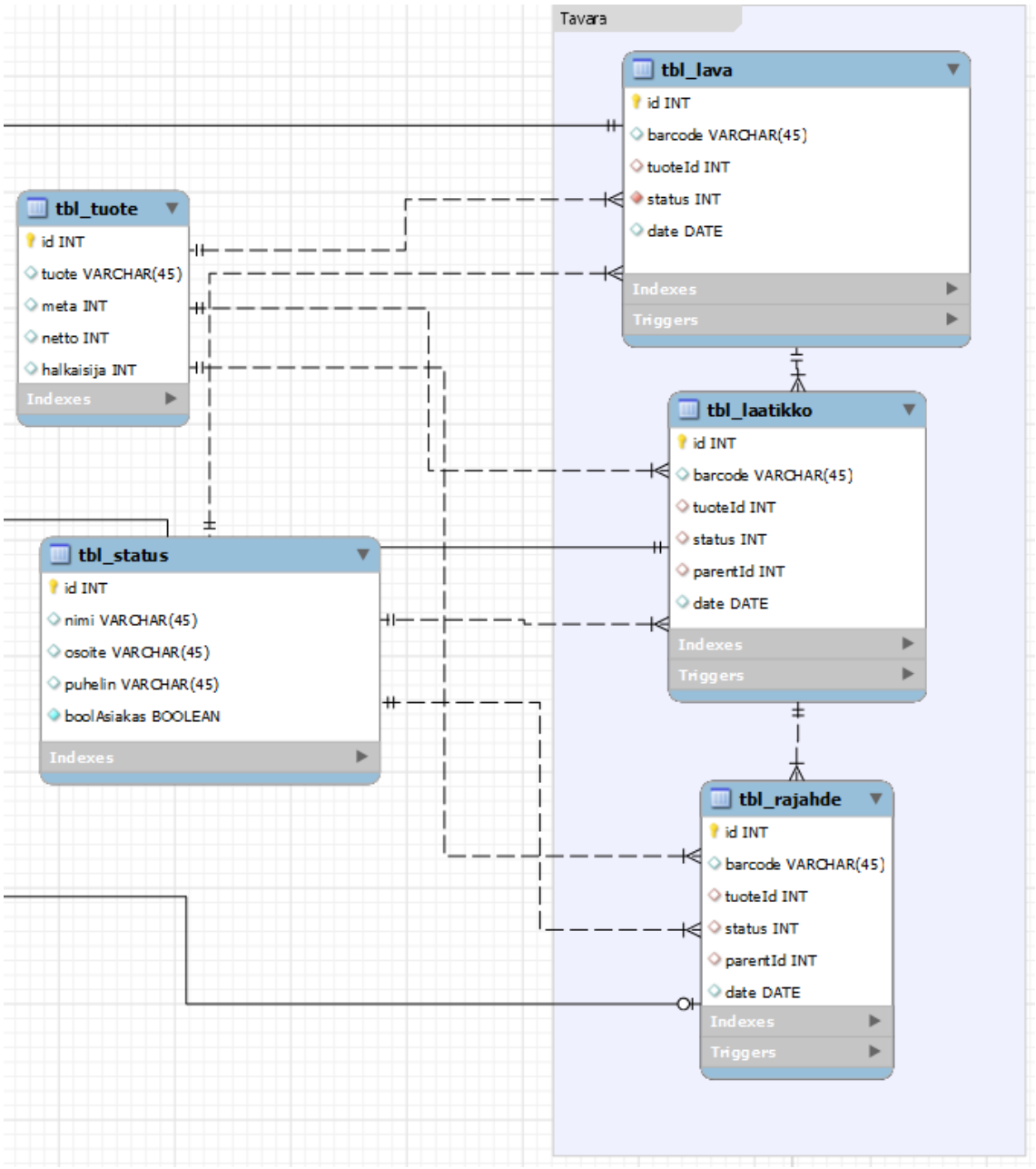
Järjestelmää lähdettiin alun perin toteuttamaan Scrum-projektimallin mukaisesti, mutta pian jouduttiin toteamaan ettei projektille voida järjestää tarvittavia resursseja määrättyjen sprinttien toteuttamiseksi. Projekti jatkui käytännössä prototyyppimenetelmällä, jossa tuotettiin ominaisuus kerrallaan johdonkatselmusta varten, jonka palautteen perusteella jatkokehitettiin prototyyppikomponenttia tai siirryttiin seuraavan tuottamiseen.

4.1 MySQL-relaatiotietokanta

Ensimmäisenä toiminnallisuutena toteutettiin seurantajärjestelmän tietokanta. Tietokantaan tallennettavat tuotteet päätettiin selkeyden vuoksi jakaa kolmeen tauluun pakkaustason mukaan. Näiden välisten suhteiden tallentamiseksi laatikko- ja patruunatason tietueilla on parentId-muuttuja, jonka arvo kertoo missä ylemmän pakkausluokan yksikössä tuote sijaitsee.

Laatikko- ja patruunatason tietueilla on parent-viiteavain ylemmän luokan tauluun, joka viittaa tunnistenumeron lisäksi myös status-muuttujaan. Näin tietokanta huolehtii tuotteen statusta muutettaessa siitä, että siihen liitettyjen lapsituotteiden status muuttuu vastaavaksi. Viiteavaimen merkitseminen tyhjäksi (*null*) on sallittu, jotta esimerkiksi toimitettaessa yksittäinen laatikko asiakkaalle voidaan tuhota yhteys isäntäyksikköön ennen statuksen päivittämistä.

Jokaisella tuotteella on kaksi muuta viiteavainta, jotka viittaavat tauluihin *tbl_tuote* ja *tbl_status*. *Tbl_tuote* pitää sisällään kaikki järjestelmään kirjatut tuotetyypit, kun taas taulussa *tbl_status* on kirjattuna kaikki tuotteiden mahdolliset sijainnit (kuva 3). Tämä tarkoittaa omien varastojen lisäksi myös kaikkia asiakkaita eli varastonseurantajärjestelmän osalta yksinkertaistettua asiakasrekisteriä.



KUVA 3. Tietokannan ER-kaavio

Yksi järjestelmän ensimmäisestä versiosta pois jätetty ominaisuus oli, että tuotteilla olisi siirtohistoria jäljityksen oikeellisuuden varmistamiseksi nykyisen statuksen lisäksi. Kuitenkin projektin loppupuolella perehdyin MySQL:n tarjoamaan trigger-toimintoon. Triggerit ovat tietokantaan tallennettavia funktioita, jotka aktivoituvat ennalta määrätyn ehdon täyttyessä. Siirtohistorian ylläpitämiseen nämä ovat erityisen käteviä, sillä työ tapahtuu palvelimella eikä asiakasohjelman tarvitse erikseen ottaa palvelinyhteyttä asiasta huolehtiakseen.

Loin jokaiselle pakkausten tallentamiseen käytetylle taululle oman triggerin joka aktivoituu tauluun kohdistuvaa UPDATE-komentoa käytettäessä. Funktio tarkistaa, onko tietueen status tai parentId muuttumassa ja mikäli on, tallentaa se erilliseen lokitauluun kyseisten muuttujien sekä vanhat että uudet arvot ja päivämäärän (kuva 4).

```
CREATE TRIGGER `tbl_laatikko_BUPD` BEFORE UPDATE ON `tbl_laatikko` FOR EACH ROW
IF NOT( NEW.status <=> OLD.status AND OLD.parentId <=> NEW.parentId) THEN
INSERT INTO tbl_laatikko_log (id, old_status, status, old_date, date, old_parent, new_parent)
VALUES( OLD.id,
OLD.status,
NEW.status,
OLD.date,
NEW.date,
OLD.parentId,
NEW.parentId);
END IF;
END
```

KUVA 4. Esimerkki trigger-funktiosta.

4.2 Viivakoodien lukeminen

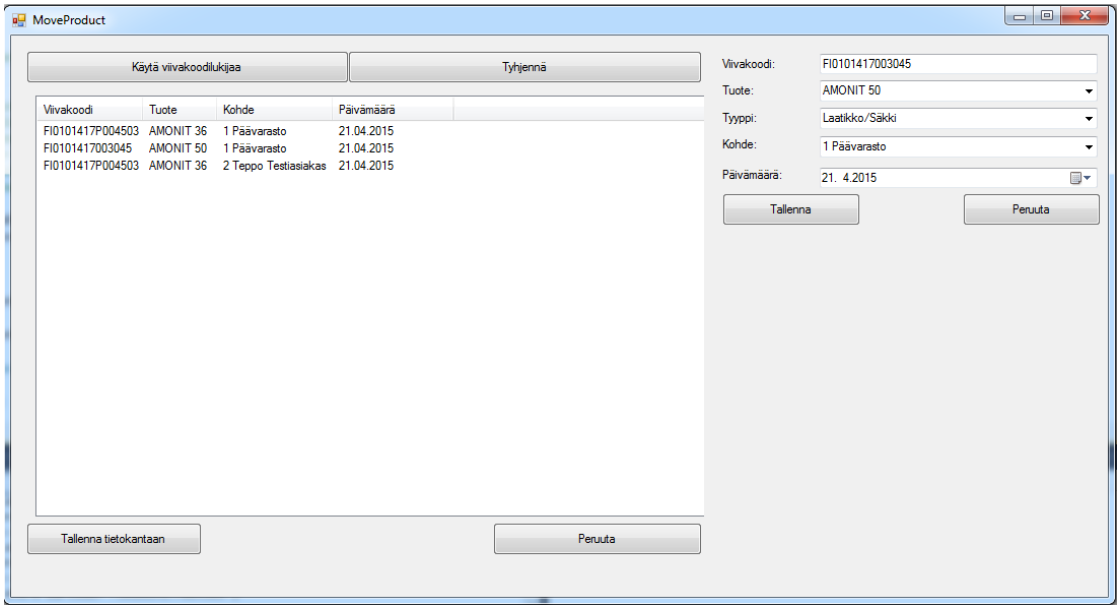
Viivakoodien keräyspäätteenä tilaajalla oli jo ennestään käytössä CipherLab 8200. Päätelaite kytketään tietokoneen USB-porttiin ja koodien purkaminen tapahtuu virtuaalisen sarjaportin (VCOM) kautta. Tietokone vastaanottaa datan tekstisyötteenä, joten viivakoodien lukemisessa täytyi kiinnittää huomiota siihen, että käyttöjärjestelmässä fokus pysyy lukemisesta vastaavalla komponentilla, jotteivat esimerkiksi välilyönnit tai sarkaimet aktivoisi tahattomasti mitään kontrolleja.

Tahattomien painallusten muodostaman ongelman havaittuani loin viivakoodien lukemista varten uutena ikkunana avautuvan Form-luokan, jossa ei ole mitään painikkeita. Lukijan syöte vastaanotetaan Formin KeyPress- ja PreviewKeyDown-eventeissä (kuva 6).

```
private void BarcodeReader_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)
{
    // Tunnistetaan lukijan erottimena käyttämät sarkainmerkit joita KeyPress-Event ei havaitse,
    // lisätään omavalintainen erotinmerkki sen tilalle.
    if (parsing && e.KeyCode == Keys.Tab) buffer += '^';
    // Lukeminen on valmis, suljetaan Form.
    if (ready) this.Close();
}

private void BarcodeReader_KeyPress(object sender, KeyPressEventArgs e)
{
    if (parsing)
    {
        this.labelInfoLabel.Text = "Luetaan koodeja...";
        // Resetoidaan ajastin aina syötettäessä merkki.
        readTimer.Stop();
        readTimer.Start();

        buffer += e.KeyChar;
        // Asetetaan merkki käsitellyksi.
        e.Handled = true;
    }

}
```

KUVA 6. Lukijan syötteen vastaanottaminen ohjelmassa.

Luetut viivakoodit näytetään omassa ikkunassa (kuva 7), jossa käyttäjän on tarvittaessa myös mahdollista syöttää koodeja käsin. Käsin muokkausta varten ohjelma hakee mahdolliset tuote- ja statustunnisteet tietokannasta ja ne ovat helposti valittavissa pudotuslaatikoista.



KUVA 7. Tuotteiden siirtonäkymä työpöytäsovelluksessa.

4.3 Tietokantakomponentti

Tietokannan käyttämiseen työpöytäsovelluksessa loin SqlConnector-luokan, jonka tehtävänä on toimia käyttöliittymän rajapintana MySQL-palvelimelle. Käyttöliittymän ohjelmakoodissa voidaan keskittyä olennaiseen eli datan käsittelyyn ja esittämiseen käyttäjälle SQL-kyselyiden muodostamisen tapahtuessa siitä erillään.

Suurin osa SqlConnectorin tarjoamista toiminnoista, oli kyseessä sitten tiedon haku tai tallentaminen, ottaa vastaan parametrina olion tai listan olioista jota voidaan käyttöliittymän kontrolleissa käyttää suoraan, esimerkiksi ListViewItemCollection. SqlConnectorin funktiot siis huolehtivat tietokannasta noudetun datan muuntamisesta käyttöliittymälle sopivaan muotoon niin, ettei käyttöliittymän koodin tarvitse huolehtia tietokantaan liittyvistä toiminnoista eikä kyseisiä muunnoksia tarvitse toistaa eri käyttöliittymään kuuluvissa komponenteissa.

4.4 Lähetyslistan lukeminen

XML-lähetyslistojen lukemiseen loin erillisen FeemXMLHandler-kirjaston, sillä tiedossa oli että samoja toiminnallisuuksia tullaan tarvitsemaan muissakin ohjelmissa.

XML-muotoisen lähetyslistan parseroimisen käymällä elementtien muodostamaa rakennetta läpi, kunnes varastonseurannan kannalta tarpeelliset tiedot on kerätty. Ensiksi ohjelma etsii jokaisen SummaryItem-tyyppisen elementin, joista voidaan lukea lähetyksen sisältämät tuotetyypit. Mikäli kyseessä on uusi tuotetyyppi, SummaryItemin nimestä parseroidaan tuotteen tyyppi ja yksikön koko. Nämä tiedot tallennetaan tietokantaan tuotetauluun.

Tämän jälkeen voidaan käydä lavat ja niiden sisältö läpi tallentaen ne tietokantaan. SummaryItemissä olevan SID-koodin (Shipment ID) avulla voidaan yhdistää kukin pakkaus tuotetyyppeihin. Jokainen pakkaus tallennetaan tietokantaan jättäen status-muuttuja tyhjäksi, joten tuotteet eivät näy varastokirjanpidossa ennen kuin ne on kirjattu vastaanotetuksi.

5 YHTEENVETO

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa varastonseurantajärjestelmä, jonka tuli varastokirjanpidon pitämisen lisäksi kyetä jäljittämään yksilöllisillä viivakoodeilla varustettujen tuotteiden toimitustiedot. Lisäksi vastuunalaisella varastonhoitajalla tuli olla etäpääsy tietokantaan viranomaisten tietopyyntöjen varalta.

Suurin ongelma järjestelmän toteutuksessa oli, ettei viivakoodien data riitä täydellisen tietokannan luomiseen. Pakkauksen viivakoodin sisällöstä ei voitu päätellä pakkauksen sisältöä tai sisällön määrää. Tämä ratkaistiin ottamalla käyttöön XML-muotoinen standardisoitu lähetyslista, joka sisältää hierarkkisessa muodossa kaikki lähetyksen yksiköt.

Lopputuloksena saatiin valmiiksi järjestelmän ensimmäinen testiversio, joka otetaan testaukseen tilaajan toiminnassa ja jota tullaan jatkokehittämään palautteen perusteella. Uusiakin lisättäviä ominaisuuksia ilmeni jo opinnäytetyön aikana useita, kuten esimerkiksi käyttöön otetun sähköisen lähetyslistan luominen tuotteita lähetettäessä.

LÄHTEET

1. FEEM European Explosives Code Structure. 2009. Federation of European Explosive Manufacturers. Saatavissa: http://feem.info/wp-content/uploads/2013/01/Guidance-Note-FEEM-European-Code-Structure-Mod.1-April-2013.pdf. Hakupäivä 27.4.2015.

2. XML. 2015. Wikipedia. Saatavissa: http://fi.wikipedia.org/wiki/XML. Hakupäivä 31.5.2015.

3. MariaDB. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/MariaDB. Hakupäivä 3.5.2015.

4. About XtraDB. 2015. MariaDB. Saatavissa: https://mariadb.com/kb/en/mariadb/about-xtradb/. Hakupäivä 3.5.2015.